NAG Library Function Document  

nag_dgeev (f08nac)

1 Purpose

nag_dgeev (f08nac) computes the eigenvalues and, optionally, the left and/or right eigenvectors for an \( n \times n \) real nonsymmetric matrix \( A \).

2 Specification

```c
#include <nag.h>
#include <nagf08.h>

void nag_dgeev (Nag_OrderType order, Nag_LeftVecsType jobvl,
                Nag_RightVecsType jobvr, Integer n, double a[], Integer pda,
                double wr[], double wi[], double vl[], Integer pdvl, double vr[],
                Integer pdvr, NagError *fail)
```

3 Description

The right eigenvector \( v_j \) of \( A \) satisfies

\[
Av_j = \lambda_j v_j
\]

where \( \lambda_j \) is the \( j \)th eigenvalue of \( A \). The left eigenvector \( u_j \) of \( A \) satisfies

\[
u_j^H A = \lambda_j u_j^H
\]

where \( u_j^H \) denotes the conjugate transpose of \( u_j \).

The matrix \( A \) is first reduced to upper Hessenberg form by means of orthogonal similarity transformations, and the \( QR \) algorithm is then used to further reduce the matrix to upper quasi-triangular Schur form, \( T \), with 1 by 1 and 2 by 2 blocks on the main diagonal. The eigenvalues are computed from \( T \), the 2 by 2 blocks corresponding to complex conjugate pairs and, optionally, the eigenvectors of \( T \) are computed and backtransformed to the eigenvectors of \( A \).

4 References


5 Arguments

1:  \textbf{order} – Nag_OrderType  \hspace{1cm} \textit{Input}

\textit{On entry:} the \textbf{order} argument specifies the two-dimensional storage scheme being used, i.e., row-major ordering or column-major ordering. C language defined storage is specified by \textbf{order} = Nag_RowMajor. See Section 3.2.1.3 in the Essential Introduction for a more detailed explanation of the use of this argument.

\textit{Constraint:} \textbf{order} = Nag_RowMajor or Nag_ColMajor.

2:  \textbf{jobvl} – Nag_LeftVecsType  \hspace{1cm} \textit{Input}

\textit{On entry:} if \textbf{jobvl} = Nag_NotLeftVecs, the left eigenvectors of \( A \) are not computed.
If jobvl = Nag_LeftVecs, the left eigenvectors of $A$ are computed.

Constraint: jobvl = Nag_NotLeftVecs or Nag_LeftVecs.

3: jobvr – Nag_RightVecsType

On entry: if jobvr = Nag_NotRightVecs, the right eigenvectors of $A$ are not computed.

If jobvr = Nag_RightVecs, the right eigenvectors of $A$ are computed.

Constraint: jobvr = Nag_NotRightVecs or Nag_RightVecs.

4: $n$ – Integer

On entry: $n$, the order of the matrix $A$.

Constraint: $n \geq 0$.

5: $a[dim]$ – double

Note: the dimension, $dim$, of the array $a$ must be at least max$(1, pda \times n)$.

The $(i, j)$th element of the matrix $A$ is stored in

\[
a[(j - 1) \times pda + i - 1] \quad \text{when } order = \text{Nag\_ColMajor};
\]
\[
a[(i - 1) \times pda + j - 1] \quad \text{when } order = \text{Nag\_RowMajor}.
\]

On entry: the $n$ by $n$ matrix $A$.

On exit: $a$ has been overwritten.

6: $pda$ – Integer

On entry: the stride separating row or column elements (depending on the value of order) in the array $a$.

Constraint: $pda \geq \max(1, n)$.

7: $wr[dim]$ – double

8: $wi[dim]$ – double

Note: the dimension, $dim$, of the arrays $wr$ and $wi$ must be at least max$(1, n)$.

On exit: $wr$ and $wi$ contain the real and imaginary parts, respectively, of the computed eigenvalues. Complex conjugate pairs of eigenvalues appear consecutively with the eigenvalue having the positive imaginary part first.

9: $vl[dim]$ – double

Note: the dimension, $dim$, of the array $vl$ must be at least

\[
\max(1, pdvl \times n) \quad \text{when } jobvl = \text{Nag\_LeftVecs};
\]
\[1 \quad \text{otherwise}.
\]

Where $VL(i, j)$ appears in this document, it refers to the array element

\[
vl[(j - 1) \times pdvl + i - 1] \quad \text{when } order = \text{Nag\_ColMajor};
\]
\[
vl[(i - 1) \times pdvl + j - 1] \quad \text{when } order = \text{Nag\_RowMajor}.
\]

On exit: if jobvl = Nag_LeftVecs, the left eigenvectors $u_j$ are stored one after another in $vl$, in the same order as their corresponding eigenvalues. If the $j$th eigenvalue is real, then $u_j = VL(i, j)$, for $i = 1, 2, \ldots, n$. If the $j$th and $(j+1)$st eigenvalues form a complex conjugate pair, then $u_j = VL(i, j) + i \times VL(i, j + 1)$ and $u_{j+1} = VL(i, j) - i \times VL(i, j + 1)$, for $i = 1, 2, \ldots, n$.

If jobvl = Nag_NotLeftVecs, $vl$ is not referenced.
10: **pdvl** – Integer  
*Input*  
*On entry:* the stride separating row or column elements (depending on the value of *order*) in the array vl.  
*Constraints:*  
if jobvl = Nag_LeftVecs, pdvl \( \geq \max(1, n) \); otherwise \( pdvl \geq 1 \).

11: **vr[dim]** – double  
*Output*  
*Note:* the dimension, *dim*, of the array *vr* must be at least \( \max(1, pdvr \times n) \) when jobvr = Nag_RightVecs; 1 otherwise.  
Where VR(i,j) appears in this document, it refers to the array element  
\[
vr[(j - 1) \times pdvr + i - 1] \quad \text{when order = Nag_ColMajor}; 
vr[(i - 1) \times pdvr + j - 1] \quad \text{when order = Nag_RowMajor}.
\]  
*On exit:* if jobvr = Nag_RightVecs, the right eigenvectors \( v_j \) are stored one after another in *vr*, in the same order as their corresponding eigenvalues. If the \( j \)th eigenvalue is real, then \( v_j = VR(i,j) \), for \( i = 1, 2, \ldots, n \). If the \( j \)th and \( (j + 1) \)st eigenvalues form a complex conjugate pair, then \( v_j = VR(i,j) + i \times VR(i,j + 1) \) and \( v_{j+1} = VR(i,j) - i \times VR(i,j + 1) \), for \( i = 1, 2, \ldots, n \).  
If jobvr = Nag_NotRightVecs, *vr* is not referenced.

12: **pdvr** – Integer  
*Input*  
*On entry:* the stride separating row or column elements (depending on the value of *order*) in the array vr.  
*Constraints:*  
if jobvr = Nag_RightVecs, pdvr \( \geq \max(1, n) \); otherwise \( pdvr \geq 1 \).

13: **fail** – NagError *  
*Input/Output*  
The NAG error argument (see Section 3.6 in the Essential Introduction).

### 6 Error Indicators and Warnings

**NE_ALLOC_FAIL**  
Dynamic memory allocation failed.  
See Section 3.2.1.2 in the Essential Introduction for further information.

**NE_BAD_PARAM**  
On entry, argument \( \langle \text{value} \rangle \) had an illegal value.

**NE_CONVERGENCE**  
The QR algorithm failed to compute all the eigenvalues, and no eigenvectors have been computed; elements \( \langle \text{value} \rangle \) to \( n \) of wr and wi contain eigenvalues which have converged.

**NE_ENUM_INT_2**  
On entry, jobvl = \( \langle \text{value} \rangle \), pdvl = \( \langle \text{value} \rangle \) and n = \( \langle \text{value} \rangle \).  
Constraint: if jobvl = Nag_LeftVecs, pdvl \( \geq \max(1, n) \); otherwise \( pdvl \geq 1 \).
On entry, \( \text{jobvr} = \langle \text{value} \rangle \), \( \text{pdvr} = \langle \text{value} \rangle \) and \( n = \langle \text{value} \rangle \).
Constraint: if \( \text{jobvr} = \text{Nag\_RightVecs} \), \( \text{pdvr} \geq \max(1, n) \);
otherwise \( \text{pdvr} \geq 1 \).

**NE\_INT**

On entry, \( n = \langle \text{value} \rangle \).
Constraint: \( n \geq 0 \).

On entry, \( \text{pda} = \langle \text{value} \rangle \).
Constraint: \( \text{pda} > 0 \).

On entry, \( \text{pdvl} = \langle \text{value} \rangle \).
Constraint: \( \text{pdvl} > 0 \).

On entry, \( \text{pdvr} = \langle \text{value} \rangle \).
Constraint: \( \text{pdvr} > 0 \).

**NE\_INT\_2**

On entry, \( \text{pda} = \langle \text{value} \rangle \) and \( n = \langle \text{value} \rangle \).
Constraint: \( \text{pda} \geq \max(1, n) \).

**NE\_INTERNAL\_ERROR**

An internal error has occurred in this function. Check the function call and any array sizes. If the call is correct then please contact NAG for assistance.

An unexpected error has been triggered by this function. Please contact NAG. See Section 3.6.6 in the Essential Introduction for further information.

**NE\_NO\_LICENCE**

Your licence key may have expired or may not have been installed correctly. See Section 3.6.5 in the Essential Introduction for further information.

7 Accuracy

The computed eigenvalues and eigenvectors are exact for a nearby matrix \( (A + E) \), where
\[
\|E\|_2 = O(\epsilon)\|A\|_2,
\]
and \( \epsilon \) is the *machine precision*. See Section 4.8 of Anderson *et al.* (1999) for further details.

8 Parallelism and Performance

\text{f08nac} is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.

\text{f08nac} makes calls to BLAS and/or LAPACK routines, which may be threaded within the vendor library used by this implementation. Consult the documentation for the vendor library for further information.

Please consult the X06 Chapter Introduction for information on how to control and interrogate the OpenMP environment used within this function. Please also consult the Users’ Note for your implementation for any additional implementation-specific information.

9 Further Comments

Each eigenvector is normalized to have Euclidean norm equal to unity and the element of largest absolute value real and positive.

The total number of floating-point operations is proportional to \( n^3 \).

The complex analogue of this function is \text{f08nnc}.
10 Example

This example finds all the eigenvalues and right eigenvectors of the matrix

\[
A = \begin{pmatrix}
0.35 & 0.45 & -0.14 & -0.17 \\
0.09 & 0.07 & -0.54 & 0.35 \\
-0.44 & -0.33 & -0.03 & 0.17 \\
0.25 & -0.32 & -0.13 & 0.11
\end{pmatrix}
\]

10.1 Program Text

/* nag_dgeev (f08nac) Example Program. */
* Copyright 2014 Numerical Algorithms Group.
* Mark 23, 2011. */

#include <stdio.h>
#include <nag.h>
#include <nag_stdlib.h>
#include <nagf08.h>

int main(void)
{
    /* Scalars */
    Integer i, j, n, pda, pdvr;
    Integer exit_status = 0;
    /* Arrays */
    double *a = 0, *vr = 0, *wi = 0, *wr = 0;
    double dummy [1];
    /* Nag Types */
    NagError fail;
    Nag_OrderType order;

    #ifdef NAG_COLUMN_MAJOR
    #define A(I, J) a[(J-1)*pda + I - 1]
    #define VR(I, J) vr[(J)*pdvr + I]
    order = Nag_ColMajor;
    #else
    #define A(I, J) a[(I-1)*pda + J - 1]
    #define VR(I, J) vr[(I)*pdvr + J]
    order = Nag_RowMajor;
    #endif

    INIT_FAIL(fail);
    printf("nag_dgeev (f08nac) Example Program Results\n");

    /* Skip heading in data file */
    #ifdef _WIN32
    scanf_s("%*[\n]");
    #else
    scanf("%*[\n]");
    #endif
    #ifdef _WIN32
    scanf("%"NAG_IFMT "%*[\n]", &n);
    #else
    scanf("%"NAG_IFMT "%*[\n]", &n);
    #endif
    if (n < 0)
    {
        printf("Invalid n\n");
        exit_status = 1;
        goto END;
    }

    /* Call nag_dgeev */
    failing = nag_dgeev(n, a, pda, vr, pdvr, wi, wr, &exit_status);
    if (failing) goto END;

    printf("roots\n");
    for (i = 0; i < n; i++)
        printf("%f
", wr[i]);

    printf("left\n");
    for (i = 0; i < n; i++)
        for (j = 0; j < n; j++)
            printf("%f
", vr[i][j]);

    printf("right\n");
    for (i = 0; i < n; i++)
        for (j = 0; j < n; j++)
            printf("%f
", A(i, j));

    printf("exit status = %d\n", exit_status);

    return 0;
}

Mark 25
pda = n;
pdvr = n;
/*@ Allocate memory */
if (!(a = NAG_ALLOC(n * n, double)) ||
   !(vr = NAG_ALLOC(n * n, double)) ||
   !(wi = NAG_ALLOC(n, double)) ||
   !(wr = NAG_ALLOC(n, double)))
{
    printf("Allocation failure\n");
    exit_status = -1;
    goto END;
}

/*@ Read the matrix A from data file */
for (i = 1; i <= n; ++i)
#include
for (j = 1; j <= n; ++j) scanf_s("%lf", &A(i, j));
#include
for (j = 1; j <= n; ++j) scanf("%lf", &A(i, j));

/*@ Compute the eigenvalues and right eigenvectors only of A */
using nag_dgeev (f08nac).

nag_dgeev(order, Nag_NotLeftVecs, Nag_RightVecs, n, a, pdar, wr, wi,
          dummy, 1, vr, pdvr, &fail);
if (fail.code != NE_NOERROR)
{
    printf("Error from nag_dgeev (f08nac).
    %s
", fail.message);
    exit_status = 1;
    goto END;
}

/*@ Print eigenvalues and right eigenvectors. */
for (j = 0; j < n; ++j)
{
    printf("\nEigenvalue %3"NAG_IFMT" = ", j+1);
    if (wi[j] == 0.0)
        printf("%13.4e\n", wr[j]);
    else
        printf(" (%13.4e, %13.4e)\n", wr[j], wi[j]);

    printf("\nEigenvector %2"NAG_IFMT"
", j+1);
    if (wi[j] == 0.0)
        for (i = 0; i < n; ++i) printf("%s%13.4e\n", "", VR(i, j));
    else if (wi[j] > 0.0)
        for (i = 0; i < n; ++i)
            printf("%s(%13.4e, %13.4e)\n", "", VR(i, j), VR(i, j+1));
    else
        for (i = 0; i < n; ++i)
            printf("%s(%13.4e, %13.4e)\n", "", VR(i, j-1), -VR(i, j));
    printf("\n");
}

END:
NAG_FREE(a);
NAG_FREE(vr);
NAG_FREE(wi);
NAG_FREE(wr);

return exit_status;

#include
#define A
#define VR

f08nac.6 Mark 25
10.2 Program Data
nag_dgeev (f08nac) Example Program Data

<table>
<thead>
<tr>
<th>A</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>: n</td>
</tr>
<tr>
<td>0.35</td>
<td>0.45</td>
</tr>
<tr>
<td>0.09</td>
<td>0.07</td>
</tr>
<tr>
<td>-0.44</td>
<td>-0.33</td>
</tr>
<tr>
<td>0.25</td>
<td>-0.32</td>
</tr>
</tbody>
</table>

: matrix A

10.3 Program Results
nag_dgeev (f08nac) Example Program Results

Eigenvalue 1 = 7.9948e-01

Eigenvector 1

-6.5509e-01
-5.2363e-01
5.3622e-01
-9.5607e-02

Eigenvalue 2 = (-9.9412e-02, 4.0079e-01)

Eigenvector 2

(-1.9330e-01, 2.5463e-01)
(2.5186e-01, -5.2240e-01)
(9.7182e-02, -3.0838e-01)
(6.7595e-01, 0.0000e+00)

Eigenvalue 3 = (-9.9412e-02, -4.0079e-01)

Eigenvector 3

(-1.9330e-01, -2.5463e-01)
(2.5186e-01, 5.2240e-01)
(9.7182e-02, 3.0838e-01)
(6.7595e-01, 0.0000e+00)

Eigenvalue 4 = -1.0066e-01

Eigenvector 4

1.2533e-01
3.3202e-01
5.9384e-01
7.2209e-01